PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-287571
(43)Date of publication of application : 31.10.1995

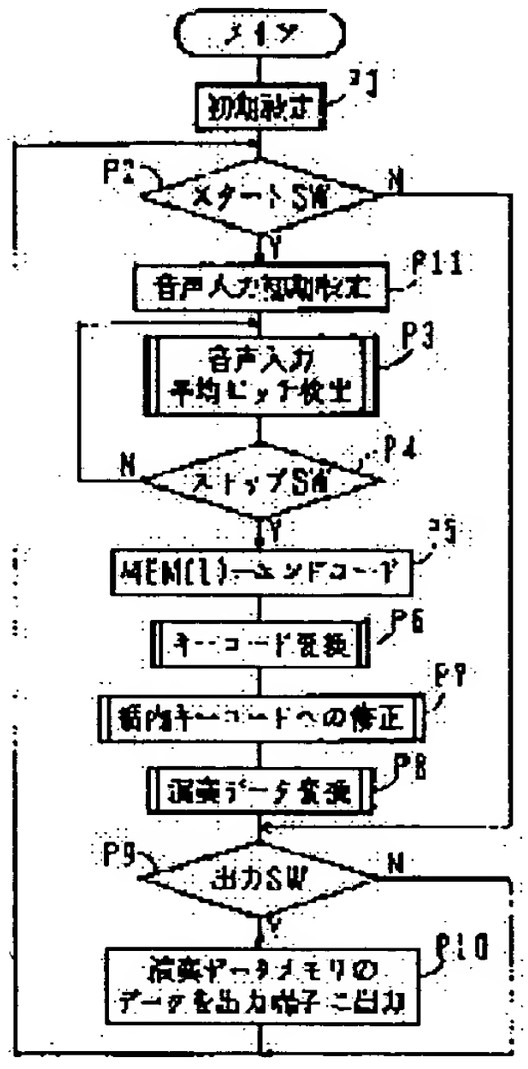
(51)Int.CI. G10H 1/00
G10L 3/00

(21)Application number : 06-081392
(22)Date of filing : 20.04.1994

(71)Applicant : YAMAHA CORP
(72)Inventor : AOKI EIICHIRO

(54) SOUND SIGNAL CONVERSION DEVICE

(57)Abstract:
PURPOSE: To convert a sound signal into a correct key code irrespective of the kind of the inputted sound signal.
CONSTITUTION: This sound signal conversion device is provided with a sound input means which inputs sound and generates the sound signal, a pitch extracting means which divides the sound signal generated by the sound input means into specific time sections and extracts pitches in the respective time sections, a mean pitch arithmetic means which calculates the mean pitch of the pitches extracted by the pitch extracting means, and a pitch information converting means which converts the mean pitch calculated by the mean pitch arithmetic means into pitch information. Further, the device has a sound input means which inputs a sound constituting music and generates a sound signal, a pitch extracting means which extracts a pitch from the sound signal generated by the sound input means, a pitch information converting means which converts the pitch extracted by the pitch extracting means into pitch information, a tone information supply mean which supplies tone information, and a pitch information correcting means which corrects the pitch information according to the tone information supplied from the tone information supply means.



LEGAL STATUS

[Date of request for examination] 27.06.1997

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number] 3055392

[Date of registration] 14.04.2000

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平7－287571

(43)公開日　平成７年(1995)10月31日

(51)Int.Cl.⁶		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ１０Ｈ	1/00	B			
Ｇ１０Ｌ	3/00	B			

審査請求　未請求　請求項の数２　ＯＬ　（全 13 頁）

(21)出願番号　特願平6－81392

(22)出願日　平成６年(1994)４月20日

(71)出願人　000004075
ヤマハ株式会社
静岡県浜松市中沢町10番１号

(72)発明者　青木　栄一郎
静岡県浜松市中沢町10番１号　ヤマハ株式会社内

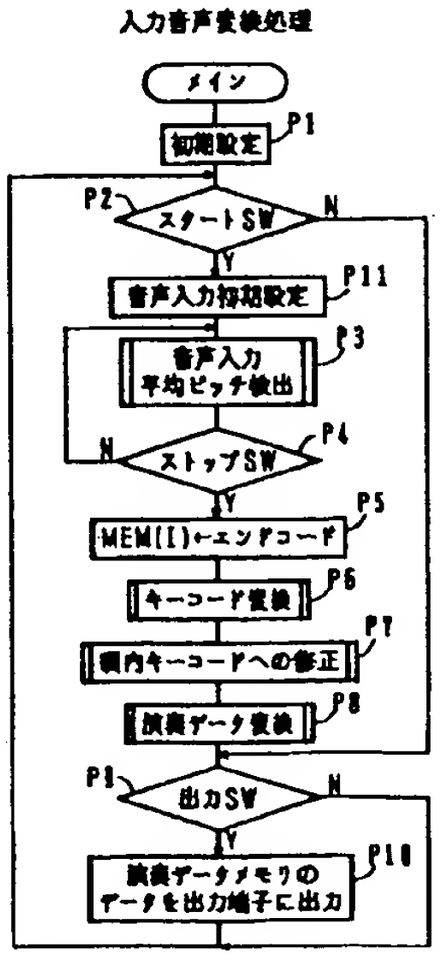
(74)代理人　弁理士　高橋　敬四郎　（外１名）

(54)【発明の名称】　音声信号変換装置

(57)【要約】

【目的】　入力される音声信号の種類にかかわらず、音声信号を正しいキーコードに変換することができる音声信号変換装置を提供することを目的とする。

【構成】　音声を入力して音声信号を生成するための音声入力手段と、音声入力手段が生成する音声信号を所定の時間区間に分割して各時間区間におけるピッチを抽出するためのピッチ抽出手段と、ピッチ抽出手段により抽出される複数のピッチを平均化して平均ピッチを演算するための平均ピッチ演算手段と、平均ピッチ演算手段により演算される平均ピッチを音高情報に変換するための音高情報変換手段とを有する。また、楽曲を構成する音声を入力して音声信号を生成するための音声入力手段と、音声入力手段が生成する音声信号からピッチを抽出するためのピッチ抽出手段と、ピッチ抽出手段により抽出されるピッチを音高情報に変換するための音高情報変換手段と、調情報を供給するための調情報供給手段と、調情報供給手段から供給される調情報に応じて音高情報を修正するための音高情報修正手段とを有する。



【特許請求の範囲】

【請求項１】　外部から供給される音声を入力して音声信号を生成するための音声入力手段と、
前記音声入力手段が生成する音声信号を所定の時間区間に分割して該各時間区間におけるピッチを抽出するためのピッチ抽出手段と、
前記ピッチ抽出手段により抽出される複数のピッチを平均化して平均ピッチを演算するための平均ピッチ演算手段と、
前記平均ピッチ演算手段により演算される平均ピッチを音高情報に変換するための音高情報変換手段とを有する音声信号変換装置。

【請求項２】　外部から楽曲を構成する音声を入力して音声信号を生成するための音声入力手段と、
前記音声入力手段が生成する音声信号からピッチを抽出するためのピッチ抽出手段と、
前記ピッチ抽出手段により抽出されるピッチを音高情報に変換するための音高情報変換手段と、
調情報を供給するための調情報供給手段と、
前記調情報手段から供給される調情報に応じて前記音高情報を修正するための音高情報修正手段とを有する音声信号変換装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は音声信号変換装置に関し、特に入力された音声信号をキーコードに変換する変換装置に関する。

【０００２】

【従来の技術】音声信号変換装置は、外部から入力される音声信号をキーコード（音高情報）に変換する装置である。変換されたキーコードは、例えばＭＩＤＩ形式の演奏データとして扱うことができる。音源装置でこのような演奏データを楽音信号に変換すれば、発音可能な信号となる。

【０００３】音声信号変換装置は、逐次入力される音声信号が何のキーコードであるのかを検出するための認識処理を行う。音声信号から正しいキーコードを認識することができる割合を示すキーコード認識率は、音声信号変換装置に入力される音声信号の種類により大きく異なる。

【０００４】入力音声信号が楽器演奏により発せられたものであれば、キーコードの認識率はある程度高い値がえられる。しかし、入力音声信号が人間の音声である場合には、楽器演奏による音声の検出に比べてキーコードの認識率が低い値になってしまう。

【０００５】人間の喉から発せられる声は、通常同一のキーコードを持続して有することは少なく、揺らいだ音になる。プロ歌手による歌声であっても、発せられる音声信号中の基本周波数はある程度揺れてしまうために、歌声から１音１音のキーコードを正確に認識することは困難である。

【０００６】

【発明が解決しようとする課題】電子楽器のように、電気信号の合成により生成される音声信号は、各周波数成分の変動が少なく雑音がほとんど含まれていない信号である。したがって、電子楽器による音声信号から対応するキーコードを認識することは比較的容易に行うことができる。

【０００７】しかし、人間の発声等により発せられる音声信号は、周波数成分の変動が大きいためにキーコードを認識することが困難である。本発明の目的は、入力される音声信号の種類にかかわらず、音声信号を正しいキーコードに変換することができる音声信号変換装置を提供することである。

【０００８】

【課題を解決するための手段】本発明の音声信号変換装置は、外部から供給される音声を入力して音声信号を生成するための音声入力手段と、音声入力手段が生成する音声信号を所定の時間区間に分割して各時間区間におけるピッチを抽出するためのピッチ抽出手段と、ピッチ抽出手段により抽出される複数のピッチを平均化して平均ピッチを演算するための平均ピッチ演算手段と、平均ピッチ演算手段により演算される平均ピッチを音高情報に変換するための音高情報変換手段とを有する。

【０００９】また、本発明の音声信号変換装置は、外部から楽曲を構成する音声を入力して音声信号を生成するための音声入力手段と、音声入力手段が生成する音声信号からピッチを抽出するためのピッチ抽出手段と、ピッチ抽出手段により抽出されるピッチを音高情報に変換するための音高情報変換手段と、調情報を供給するための調情報供給手段と、調情報供給手段から供給される調情報に応じて音高情報を修正するための音高情報修正手段とを有する。

【００１０】

【作用】ピッチ抽出手段は、音声信号入力手段に入力された音声信号を所定の時間区間に分割して、各時間区間毎のピッチを検出する。検出されたピッチは、平均ピッチ演算手段により複数の時間区間で平均化された後に、音高情報に変換される。音高情報は、検出されたピッチを直接用いるよりも、平均化したピッチを用いた方が正しい音高情報を得ることができる。

【００１１】また、楽曲を構成する入力音声信号から検出されたピッチを音高情報に変換した後に、音高情報は、供給された調情報の音楽規則に応じて修正される。音高情報の修正を行うことにより、正しい音高情報を得ることができる。

【００１２】

【実施例】図１は、本発明を実施するための音声信号変換装置のシステム構成例を示す。音声信号変換装置は、マイク１から入力される音声信号をキーコードに変換す

ることができる。変換されたキーコードは、音源4において対応するキーコードの楽音信号に変換され、サウンドシステム5から発音される。

【0013】マイク1には、外部から歌声等の音声信号が入力される。入力される音声信号は、例えば人の歌声や楽器演奏等による楽音のアナログ信号である。スタートスイッチを押すと、音声信号の入力が開始され、ストップスイッチを押すと、音声信号の入力が終了する。マイク1に入力された音声アナログ信号は、D／A変換器2においてディジタル信号に変換され、バス11に供給される。

【0014】CPU7は、バス11に供給された音声ディジタル信号をキーコードに変換するために、各種の制御処理を行う。正しいキーコードに変換するために、CPU7は通常のキーコード変換処理の他に、平均ピッチ検出と調内キーコードへの修正処理を行う。

【0015】平均ピッチ検出は、マイク1に入力された音声信号を量子化して、量子化された音声信号毎にピッチを検出する。そして、所定時間内で検出された複数のピッチの平均ピッチを求め、平均ピッチからキーコードを算出する。平均ピッチを検出することにより、人の歌声のように時間経過と共にピッチが変化しやすい性質の音声信号であっても、正しいピッチを検出することができる。

【0016】調内キーコードへの修正処理は、まず、連続してマイク1に入力された音声信号を楽曲とみなして、その楽曲の調を検出する。楽音の調が検出されれば、その調に用いられやすいキーコードを予測することができる。この音楽規則の知識を利用して、検出されたキーコードの修正処理を行う。これにより、誤認識により検出されたキーコードを正しいキーコードに修正することができる。

【0017】データメモリ（RAM）10は、入力音声信号を楽音とみなしたときの調を検出するために必要なデータ等を記憶している。例えば、C長調、C短調等の各調を構成する音名データを記憶しており、入力音声信号のキーコードがどの調の構成音に該当するのかを調べることにより、調の検出が行われる。

【0018】CPU7は、入力音声信号からキーコードに変換した後に、変換したキーコードを基にして演奏データを生成する。生成された演奏データは音源4に供給され、楽音信号が生成される。サウンドシステム5は、音源4から供給される楽音信号に応じて楽音を発する。

【0019】また、出力端子3の出力スイッチが押されると、生成された演奏データは、必要に応じてMIDI形式のデータ等に変換され、出力端子3から外部機器に出力される。

【0020】プログラムメモリ（ROM）8は、演算プログラムを記憶している。CPU7はこの演算プログラムに従って、ワーキングメモリ（RAM）9に備えられ

たレジスタやバッファメモリ等のワーキングメモリを用いて上述の各種演算処理を行う。

【0021】タイマ6は、タイミング信号を生成し、所定時間間隔でCPU7に割り込み信号を供給する。タイマ6は、例えば16分（音符）の長さ毎に割り込み信号をCPU7に供給し、割り込み処理を実行させる。割り込み処理は、外部から人がマイク1に歌声を入力する際等に必要となるメトロノーム音を生成することもできる。

【0022】CPU7は、バス11を介して、D／A変換器2、プログラムメモリ8、ワーキングメモリ9、データメモリ10、音源4、出力端子3の制御を行う。図2は、CPUが行う入力変換処理のメインルーチンを示すフローチャートである。CPUは、まずステップP1において、レジスタ類の初期化等の初期設定の処理を行う。

【0023】ステップP2では、マイクから音声信号の入力開始を指示するスタートスイッチが押されたか否かを調べる。スタートスイッチが押されていなければ、音声入力の処理を行わずにステップP9へ進む。

【0024】スタートスイッチが押されていれば、ステップP11へ進み、音声入力処理のために必要なレジスタやフラグ等の初期設定を行い、ステップP3へ進む。ステップP3では、音声入力処理と平均ピッチ検出処理を行う。音声入力処理は、マイクから入力される音声信号をワーキングメモリに取り込む処理である。平均ピッチ検出は、入力された音声信号を量子化して、量子化された音声信号毎のピッチの平均値を検出する処理である。処理の詳細は、後にフローチャートを参照しながら説明する。

【0025】ステップP4では、マイクから音声信号の入力終了を指示するストップスイッチが押されたか否かを調べる。ストップスイッチが押されていなければ、マイクに逐次入力される音声信号について、上述の音声入力処理と平均ピッチ検出処理を繰り返し、ストップスイッチが押されるまで処理を繰り返す。

【0026】ストップスイッチが押されると、ステップP5へ進み、音声信号の入力終了を示すエンドコードをワーキングメモリ中のメモリMEM（）に格納する。上述のステップで検出された所定時間毎の平均ピッチは、既に配列メモリMEM（）に格納されている。入力された音声信号の全ての平均ピッチが格納された後に、ストップスイッチが押されると、最後の配列のメモリMEM（I）にエンドコードが格納される。

【0027】ステップP6では、メモリMEM（）に格納された音声信号の平均ピッチをキーコードに変換する処理を行う。平均ピッチは、周波数で表されているので、その周波数で表される値をC4等で表されるキーコードに変換する。詳細は、後述する。

【0028】ステップP7では、変換されたキーコード

の中で誤って変換されたキーコードの修正を行うための処理を行う。まず、前ステップにおいて得られた複数のキーコードを楽曲とみなして、その楽音の調を検出する。そして、検出された調の構成音に該当しないものについては、誤ったキーコードに変換されたと判断して、修正処理を行う。処理の詳細は、後にフローチャートを参照しながら説明する。

【００２９】ステップＰ８では、キーコードの検出および修正が行われた後に、そのキーコードを基にして演奏データの生成を行う。演奏データは、キーコードの他に、発音開始を示すキーオンコード、発音の持続時間を示すデュレーションコード、発音終了を示すキーオフコード等のデータを含んでいる。詳細は、後述する。

【００３０】ステップＰ９では、出力スイッチが押されたか否かを調べる。出力スイッチが押されていれば、ステップＰ１０へ進み、生成された演奏データを出力端子から出力する。その後、ステップＰ２へ戻り、再び音声入力開始のためのスタートスイッチが押されたか否かを調べる。

【００３１】ステップＰ９において、出力スイッチが押されていないと判断されたときには、出力端子から演奏データを出力することなく、直接ステップＰ２へ戻る。図３は、図２のステップＰ３における音声入力処理と平均ピッチ検出処理の詳細を示すフローチャートである。

【００３２】図２のメインルーチンにおいて、この音声入力処理が行われる前に、ステップＰ１１で音声入力処理のための初期設定が既に行われている。初期設定では、キーオンフラグＫＯＮ、ピッチレジスタＦ、フラグＱ、カウンタＩ、カウンタＮの初期化を行っている。各フラグ等の初期設定方法を次に示す。

【００３３】まず、キーオンフラグＫＯＮを０にする。キーオンフラグＫＯＮは、マイクから何らかの音声信号が入力されている間は１を示し、入力される音声信号がほぼ０である間は０を示す。

【００３４】次に、ピッチレジスタＦを０にクリアする。ピッチレジスタＦは、マイクから入力される音声信号から検出されるピッチを格納するためのレジスタである。ピッチとは、音高を表す周波数である。

【００３５】そして、フラグＱを０にする。フラグＱは、タイマからの割り込み信号をきっかけとする割り込み処理により、所定時間（例えば１６分音符の長さ）間隔で１にセットされる。フラグＱは、所定の処理が行われた後に０になるので、通常は０がセットされており、所定時間間隔で一瞬１がセットされる。

【００３６】さらに、カウンタＩおよびカウンタＮを０にリセットする。カウンタＩは、求められる平均ピッチを格納する配列メモリＭＥＭ（）の配列番号を示す。カウンタＮは、所定時間内に入力される音声信号の量子化数をカウントする。

【００３７】以上の初期設定が行われた後に、音声入力

および平均ピッチ検出を行うための処理がステップＱ２から開始する。ステップＱ２では、キーオンフラグＫＯＮが０であるか否かを調べる。キーオンフラグＫＯＮが０であれば、ステップＱ３に進む。最初は、初期設定でキーオンフラグＫＯＮが０に設定されているので、ステップＱ３に進む。

【００３８】ステップＱ３では、マイクから入力される音声信号の入力レベルがしきい値＃１を越えているか否かを調べる。しきい値＃１は、音声入力レベルの雑音レベルを示す。しきい値＃１を越えていなければ、入力が無音状態であると判断して、入力音声信号の処理を行わずにステップＱ１０へ進む。

【００３９】音声入力レベルがしきい値＃１を越えていれば、適正な音声入力があったとして、ステップＱ４へ進み、入力音声信号の処理を開始する。ステップＱ４では、適正な音声入力の開始を示すために、キーオンフラグＫＯＮを１にする。その後、ステップＱ５へ進む。

【００４０】ステップＱ５では、入力音声信号を基にしてピッチ検出を行う。ディジタル信号系列で表される入力音声信号に対して、ハミング窓やハニング窓等で所定時間内の信号を切り出し、周波数解析を行う。周波数解析を行うことにより、ピッチ（音高情報）を検出する。

【００４１】ステップＱ６では、検出されたピッチをピッチレジスタＦに加算する。ピッチレジスタＦには、初期設定で０にセットされているので最初は、検出されたピッチそのものの値がピッチレジスタＦに格納される。

【００４２】ステップＱ７では、カウンタＮをインクリメントする。その後、ステップＱ１０へ進む。ステップＱ１０では、フラグＱが１であるか否かを調べる。フラグＱは、初期設定において０にセットされているので、一旦図２のメインルーチンの処理へ戻る。もし、音声入力の終了を指示するストップスイッチが押されていなければ、再び上述の音声入力処理を繰り返すためにステップＱ２からの処理を再開する。ストップスイッチが押されるまで、音声入力処理を繰り返すことにより、マイクから逐次入力される音声信号を時間経過と共にリアルタイムに処理することができる。

【００４３】音声入力処理を再開するために、ステップＱ２において、再びキーオンフラグＫＯＮが０であるか否かを調べる。既に上述の１回目のステップＱ４の処理でキーオンフラグＫＯＮが１にセットされているので、ステップＱ８へ進む。

【００４４】ステップＱ８では、今回マイクから入力された音声入力レベルがしきい値＃２よりも小さいか否かを調べる。しきい値＃２は、音声入力レベルの雑音レベルを示す。しきい値＃２を越えていれば、前回から適正な音声信号が引き続き入力されていることを示す。一方、しきい値＃２を越えていなければ、マイクからの入力が無音状態になったことを示す。

【００４５】ステップＱ８において、音声入力レベルが

しきい値＃２を越えていればマイクから音声入力が引き続き行われていることを示すので、ステップＱ５へ進み、今回入力された音声信号の平均ピッチ検出の処理を行う。

【００４６】ステップＱ５では、入力音声信号のピッチを検出し、検出したピッチをステップＱ６においてピッチレジスタＦに加算する。ステップＱ７では、カウンタＮをインクリメントする。

【００４７】ストップスイッチが押されるまでの間は、ピッチレジスタＦにマイクから逐次入力される音声信号のピッチが累算されていく。ピッチが累算される度にカウンタＮはインクリメントされるので、カウンタＮにはピッチレジスタＦにピッチが累算された数が格納される。

【００４８】その後ステップＱ１０において、フラグＱが１か否かを調べる。フラグＱは、次に説明する割り込み処理により値が決定される。図４は、ＣＰＵが行う割り込み処理を示すフローチャートである。ＣＰＵは、タイマから例えば１６分長（１６分音符の時間長）の所定の時間間隔で割り込み信号を受ける。ＣＰＵは、割り込み信号を受けると以下の割り込み処理を行う。ここで、割り込み処理は１６分長間隔で行われるとする。

【００４９】まず、ステップＲ１においてフラグＱを１にセットする。そして、ステップＲ２において、クリック音を発生する。その後、割り込み処理を終了して、元の音声入力の処理を再開する。

【００５０】以上のように、割り込み処理では１６分長間隔でフラグＱを１にセットして、クリック音を発生する。クリック音は、メトロノームの役割を果たす。マイクから音声信号を入力する際には、１６分長毎に鳴るクリック音のテンポに合わせて入力を行えばよい。

【００５１】図３のステップＱ１０では、１６分長が経過した時にフラグＱが１であると判断され、ステップＱ１１へ進む。ステップＱ１１では、カウンタＮが０より大きいか否かを調べる。カウンタＮには、１６分長の間に量子化してピッチを検出した数が格納されている。もし、マイクからの入力音声信号が１６分長の間ずっと無音状態であると判断されたときには、カウンタＮは０を示す。

【００５２】上述のように、マイクに適正な音声入力がされている際、カウンタＮには１以上の数が格納されているので、ステップＱ１２へ進む。ステップＱ１２では、次式により平均ピッチを求め、平均ピッチレジスタＡＦに格納する。

【００５３】ＡＦ＝Ｆ／Ｎ

ピッチレジスタＦには、１６分長の間に検出されたピッチの累算値が格納されている。ピッチの累算値Ｆを量子化数Ｎで割ることにより、ピッチの平均値ＡＦを求めることができる。その後、ステップＱ１４へ進む。

【００５４】一方、ステップＱ１１において、１６分長の間無音の音声入力状態が続いたときにはカウンタＮが０であるので、ステップＱ１３へ進む。ステップＱ１３では、平均ピッチレジスタＡＦにオール１を格納する。その後、ステップＱ１４へ進む。オール１とは、レジスタを構成するビットを全て１にするような値を示す。平均ピッチレジスタＡＦにオール１が格納されていれば、１６分長の無音状態である１６分休符がマイクから入力されたことになる。

【００５５】ステップＱ１４では、求められた平均ピッチＡＦを配列メモリＭＥＭ（Ｉ）に格納する。カウンタＩは、初期設定にて０に設定されているので、平均ピッチＡＦは０番目の配列番号のメモリＭＥＭ（０）に格納される。カウンタＩは、求められた平均ピッチの番号を示し、時間経過と共に０から順番に増加していく。

【００５６】ステップＱ１５では、カウンタＩをインクリメントして、次の平均ピッチの配列番号をセットする。ステップＱ１６では、ピッチレジスタＦ、カウンタＮ、フラグＱのそれぞれに０をセットして、次の平均ピッチを求めるための初期設定を行う。その後、図２のメインルーチンの処理に戻り、ストップスイッチが押されるまで、以上の平均ピッチを求める処理を繰り返し行う。

【００５７】次に、マイクからの入力音声が無音状態になったときの処理を説明する。ステップＱ２において、前回時に適正な音声入力があれば、キーオンフラグＫＯＮは１になっているので、ステップＱ８へ進む。音声入力レベルがしきい値＃２よりも小さい値になっていれば、ステップＱ８で無音状態であると判断され、ステップＱ９へ進む。ステップＱ９では、キーオンフラグＫＯＮを０にして、ピッチ検出を行わないでステップＱ１０へ進む。ステップＱ１０では、前述のように１６分長毎の平均ピッチを求める処理を行う。

【００５８】以上のように、マイクから適正な音声入力があったときには、１６分長の範囲内で入力音声信号の量子化を行い、量子化された信号毎のピッチを検出する。そして、検出されたピッチの累算値Ｆを量子化数Ｎで割ることにより、平均ピッチＡＦを求める。

【００５９】人の歌声等は、音声信号の揺れが大きいので、音声信号の局所的なピッチを検出するとピッチが頻繁に変化し、ピッチの誤認識が生じやすい。そこで、入力された音声信号を量子化して、量子化毎のピッチを検出した後に平均ピッチを求めることにより、正しいピッチを検出することができる。求められた平均ピッチは配列メモリＭＥＭ（）に格納される。

【００６０】図５は、メモリＭＥＭ（）の構成例を示す。メモリＭＥＭ（）には、時間経過と共に求められた平均ピッチが順番に格納されている。メモリ領域１５には、求められた平均ピッチを示す周波数が格納されている。平均ピッチは、１６分長の間に適正な音声入力があったときに求められるピッチの平均値である。

【００６１】メモリ領域１６には、オール１のデータが格納されている。オール１のデータは、無音の音声入力が１６分長の間中続いたときに格納されるデータであり、１６分休符を示す。

【００６２】メモリMEM () に格納されている周波数は、１６分音符のピッチを示す周波数である。ただし、オール１が格納されているときには、１６分音符のピッチを示すのではなく１６分休符を意味する。

【００６３】エンドコード１７は、ストップスイッチが押されて、全ての音声入力が終了した後に、図２のステップＰ５においてメモリMEM () 中の最後の領域に格納される。エンドコード１７は、メモリMEM () に連続して格納されている平均ピッチのデータの終了であることを示す。

【００６４】図６は、図２のメインルーチン中のステップＰ６で行うキーコード変換の処理の詳細を示すフローチャートである。キーコード変換は、メモリMEM () に格納されている平均ピッチの周波数からキーコードへの変換を行う処理である。

【００６５】ステップＳ１では、平均ピッチ周波数が格納されているメモリMEM () の中において最低の周波数を示す平均ピッチを抽出し、最低周波数レジスタＦｍｉｎにその最低周波数を格納する。

【００６６】ステップＳ２では、メモリMEM () に格納されている周波数を読み出し、レジスタＦに格納する。そして、次式の計算により相対キーコードＫＣを求める。

$KC = 12 \log_2 (F/Fmin)$

相対キーコードＫＣは、通常のセントの１／１００の値を示すものであり、最低周波数Ｆｍｉｎを基準にして、周波数Ｆがどれだけ高いのか又は低いのかを示す。周波数Ｆが最低周波数Ｆｍｉｎと同じ値であれば相対キーコードＫＣは０になり、周波数Ｆが最低周波数Ｆｍｉｎよりも半音高ければ相対キーコードＫＣは１になる。

【００６７】求められた相対キーコードＫＣは、周波数が元々格納されていたメモリMEM () に再び格納される。同様にして、メモリMEM () に格納されている全ての周波数を相対キーコードＫＣに変換する。メモリMEM () には、最低周波数をＦｍｉｎを基準とした１／１００のセント表現が格納される。

【００６８】ステップＳ３では、最低周波数Ｆｍｉｎをその周波数に最も近いキーコードに変換し、最低周波数のキーコードをレジスタＫＣｍｉｎに格納する。例えば、周波数が４４０〔Ｈｚ〕であったとすればＡ４のキーコードに変換して、レジスタＫＣｍｉｎに格納する。

【００６９】ステップＳ４では、メモリMEM () に格納されている相対キーコードを読み出し、レジスタＫＣに格納する。そして、次式の計算によりキーコードＫを求める。

【００７０】Ｋ＝ＫＣ＋ＫＣｍｉｎ

キーコードＫは、基準となる最低キーコードＫＣｍｉｎに相対キーコードＫＣを加算することによりえられる絶対キーコードである。

【００７１】求められたキーコードＫは、相対キーコードが元々格納されていたメモリMEM () に再び格納される。同様にして、メモリMEM () に格納されている全ての相対キーコードを絶対キーコードＫに変換する。メモリMEM () には、１６分長毎のキーコードが格納される。その後、図２のメインルーチンの処理に戻る。

【００７２】図７は、図２のメインルーチン中のステップＰ７の調内キーコードへの修正処理の詳細を示すフローチャートである。ステップＴ１では、マイクから連続して入力される音声信号の集合を楽曲とみなして、その楽曲の調を検出する。データメモリには、調の構成音を示す音名データが格納されている。

【００７３】図８は、データメモリに格納されている調の構成音の音名データを示す。調は、１２音名（Ｃ，Ｃ＃，・・・，Ｂ）をそれぞれ主音としたときの長調および短調がそれぞれあり、それぞれの調についてのデータがデータメモリに格納されている。なお、より細かい調に分けて格納してもよい。

【００７４】例えば、Ｃ長調のメモリ領域２１には、Ｃ長調の構成音であるＣ，Ｄ，Ｅ，Ｆ，Ｇ，Ａ，Ｂのコードが格納されている。同様に、Ｃ＃長調のメモリ領域２２には、Ｃ＃長調の構成音のコードが格納され、Ｃ短調のメモリ領域２３には、Ｃ短調の構成音のコードが格納されている。

【００７５】調検出は、メモリMEM () に格納されているキーコードが上記のデータメモリに記憶されている調の中でどの調に一番合致するかを調べる。メモリMEM () のキーコードとデータメモリ中の調の構成音を比較して、最も一致度の高い調を入力音声信号の調とする。

【００７６】主音レジスタＴＮには、検出された調の主音（Ｃ，Ｃ＃，Ｄ等）を格納する。モードレジスタＭＤには、検出された調のモード（長調または短調）を格納する。調検出が行われた後、ステップＴ２へ進む。

【００７７】なお、調検出の方法は、上述の方法に限られず他の方法により調検出を行ってもよい。ステップＴ２では、検出された調を用いて、キーコードを構成音に丸め込む処理を行う。入力音声信号に対応するキーコードが検出された調の構成音ではない場合、キーコードを構成音のキーコードに正す。処理の詳細な説明を次に示す。

【００７８】図９は、図７のステップＴ２の調内音丸め込み処理の詳細を示すフローチャートである。まず、基準となる楽曲の最後の音の丸め込みを行い、その後に残りの音の丸め込みを行う。その処理手順を以下に説明する。

【００７９】ステップＵ１では、メモリMEM () のｍ

番目（メモリMEM（）のキーコードに順についている、１番最後に格納されているキーコードの番号）のキーコードを読み出す。

【００８０】ステップＵ２では、読み出したキーコードが調内音（検出された調の構成音）であるか否かを調べる。調内音であれば、正しく検出されたキーコードであると判断して、ステップＵ６へ進み、読み出したキーコードそのものをレジスタＫＣに格納する。その後、ステップＵ７へ進む。

【００８１】一方、読み出したキーコードが調内音でなければ、誤認識により検出されたキーコードであると判断して、ステップＵ３へ進む。ステップＵ３では、読み出されたm番目のキーコードを調内音に丸め込む処理を行う。まず、読み出されたキーコードをそのキーコードに最も近い調内音に変換する。もし、同じ位の近さの調内音が複数あれば、次の優先順位に従った調内音に変換する。

【００８２】

１度＞５度＞３度＞２度＞６度＞７度＞４度

例えば、検出された調がC長調であるとすると１度の音がCであり、５度の音がGであり、３度の音がEである。１度の音が最優先であり、続いて５度、３度、・・・の順の調内音に変換される。

【００８３】楽曲の最後の音は、一般的にその調の１度の音で終わり易いという規則がある。したがって、１度の音を最優先し、続いて最後の音に成りやすい５度、３度、・・・の順の音を優先させて変換を行う。そして、ステップＵ４において、調内音に変換されたキーコードをレジスタＫＣに格納する。

【００８４】ステップＵ５では、レジスタＫＣに格納されているキーコードを元のキーコードを読み出した配列番号のメモリMEM（）に記憶し直す。レジスタＫＣには、調内音に修正されたキーコードが格納されている。ただし、検出されたキーコードが調内音であるときには、正しい認識が行われたとして修正を行わずに検出されたキーコードそのものがレジスタＫＣに格納されている。

【００８５】ステップＵ７では、番号レジスタmをデクリメントする。番号mの値を１小さくすることにより、時間的に１つ前の音を表す番号mに変えることができる。番号mが示す１つ前の音についての処理を次に行う。

【００８６】ステップＵ８では、メモリMEM（）に格納されているm番目のキーコードを読み出す。読み出されたキーコードは最後から２番目の音である。ステップＵ９では、読み出したキーコードが調内音であるか否かを調べる。調内音であれば、正しく検出されたキーコードであると判断して、メモリMEM（）のm番目のキーコードを修正せずにステップＵ１４へ進み、レジスタＫＣにキーコードを格納した後にステップＵ１３へ進む。

【００８７】一方、読み出したキーコードが調内音でなければ、誤認識により検出されたキーコードであると判断して、ステップＵ１０へ進む。ステップＵ１０では、読み出したm番目のキーコードを調内音に丸め込む処理を行う

ステップＵ１０では、読み出したキーコードをそのキーコードに最も近い調内音に変換する。もし、同じ位の近さの調内音が複数あるときには、レジスタＫＣに格納されているキーコードに最も近い調内音に変換する。

【００８８】レジスタＫＣには、ステップＵ４またはステップＵ６のいずれかで格納されたキーコードが記憶されている。つまり、楽曲の最後の音の修正後のキーコードが記憶されている。その最後の音のキーコードに最も近いキーコードに優先的に変換される。

【００８９】ステップＵ１１では、調内音に修正変換されたキーコードをレジスタＫＣに格納し、ステップＵ１２でレジスタＫＣに格納されたキーコードをメモリMEM（）の元の記憶場所に記憶し直す。

【００９０】ステップＵ１３では、番号mが１であるか否かを調べる。番号mが１であるということは、楽曲の１番最初の音（最初にマイクから入力された音声信号の音）についての調内音への修正処理を終了したことを示す。修正処理は、楽曲の最後の音から最初の音まで順番に行っているので、最初の音についての処理が終了すれば、全ての音についての処理が終了したことになる。

【００９１】番号mが１でなければ、ステップＵ７へ戻り、番号mをデクリメントして１つ前の音のキーコードについての修正処理を繰り返す。メモリMEM（）のm番目のキーコードを読み出し、読み出したキーコードが調内音であればキーコードの修正を行わない。読み出したキーコードが調内音でなければ、最も近い調内音のキーコードに変換を行う。近い調内音が複数あるときには、１つ前の音の修正後のキーコードに近いキーコードに優先的に変換する。

【００９２】楽曲の最後の音の優先順位は、前述のように調の１度、５度の音等を基準にして決めたが、最後の音以外の音の優先順位は、１つ前の音のキーコードに近い音に優先して、キーコードを決める。最後の音のみは、調に関連した音になり易いからである。

【００９３】以上の修正処理により、誤認識されたキーコードが修正されるとメモリMEM（）に格納される。修正処理が終了すると、図７の元の処理に戻る。なお、以上のキーコードの修正処理では、全ての調外音を調内音に修正する場合について述べたが、必ずしも全てを修正する必要はない。例えば、１小節内でのみ効力を有する♯，♭等の臨時記号を用いて部分的に調外音を用いたい場合もある。そのような場合には、検出された調において特に使用が禁止されている音（音名）のみを禁止されていない音に修正するようにしてもよい。

【００９４】また、調を用いてキーコードを修正する他

に、スケールの検出を行い使用する音高の幅を複数検出し、検出された音高の幅より外の音については所定の音高幅内の音に修正する処理を行ってもよい。スケールは調性と和音によって定まり、スケールによって使用される音名が決まるので、この音名に対応する音高にそれぞれ幅を持たせることにより、使用される音高の幅が求められる。

【００９５】図１０は、図２のメインルーチン中のステップＰ８の演奏データ変換処理の詳細を示すフローチャートである。演奏データ変換は、入力音声信号から検出されたキーコードを基にして、発音可能な演奏データ形式への変換を行う。

【００９６】ステップＶ１では、配列番号レジスタＰ、デュレーションレジスタＤ、キーオンフラグＫＯＮにそれぞれ０をセットする。配列番号レジスタＰは、キーコードが格納されているメモリＭＥＭ（）の配列番号を示す。デュレーションレジスタＤは、同一のキーコードを発音し続ける持続時間または発音を行わない休符状態の時間を格納する。キーオンフラグＫＯＮは、何らかのキーコードを発音中であるときに１となり、いずれのキーコードをも発音しないときには０となる。

【００９７】ステップＶ２では、演奏データメモリ中の書き込み位置を示す書き込みポインタを演奏データメモリの先頭にセットし、最初の演奏データの書き込みの準備を行う。

【００９８】ステップＶ３では、メモリＭＥＭ（Ｐ）にエンドコードが格納されているか否かを調べる。配列番号Ｐは、初期設定で０に設定されている。メモリＭＥＭ（）には、入力音声信号から検出された後に必要に応じて修正が行われたキーコードが格納されている。もし、メモリＭＥＭ（）にエンドコードが格納されていれば、その配列番号の位置で音声入力が終了していることを示す。

【００９９】メモリＭＥＭ（Ｐ）にエンドコードが格納されていなければ、ステップＶ４へ進み、メモリＭＥＭ（Ｐ）に格納されているキーコードを基にして演奏データに変換する。配列番号Ｐについての演奏データ変換方法は、後に説明する。

【０１００】メモリＭＥＭ（Ｐ）のキーコードについて演奏データが生成された後には、ステップＶ５へ進み、配列番号レジスタＰをインクリメントして次の配列番号についての演奏データの生成に備える。また、デュレーションレジスタＤをインクリメントして発音の持続時間をカウントする。デュレーションレジスタＤが１増えると、１６分長の長さだけ持続時間が増加する。メモリＭＥＭ（）には、１６分長毎にキーコードが格納されているからである。

【０１０１】その後、ステップＶ３へ戻り、次の配列番号Ｐに格納されているキーコード（ＭＥＭ（Ｐ））がエンドコードであるか否かを調べる。エンドコードでなければ、ステップＶ４へ進み、メモリＭＥＭ（Ｐ）に格納されているキーコードを基にして演奏データの生成を行う。そして、ステップＶ５において、配列番号レジスタＰとデュレーションレジスタＤをそれぞれインクリメントする。

【０１０２】デュレーションレジスタＤには、同一のキーコードが連続してメモリＭＥＭ（）に格納されている数が記憶される。例えば、Ａ４のキーコードが１つだけ現れて次にはキーコードが変わってしまうときには、デュレーションレジスタＤに１が格納され、Ａ４の１６分音符を示す。また、Ａ４のキーコードが２つ連続して続けば、デュレーションレジスタＤには２が格納され、Ａ４の８分音符を示す。

【０１０３】ステップＶ３において、メモリＭＥＭ（Ｐ）にエンドコードが格納されていると判断されたときには、キーコードの終了を示すので、ステップＶ６へ進む。ステップＶ６では、キーオンフラグＫＯＮが１であるか否かを調べる。キーオンフラグＫＯＮが１であれば、現在いずれかのキーコードを発音中であることを演奏データが示しているので、ステップＶ７へ進む。

【０１０４】ステップＶ７では、現在発音中のキーコードについての発音持続時間を示すデュレーションレジスタＤの値をデュレーションコードと共に演奏データメモリに書き込む。デュレーションコードは、演奏データメモリにおいて、発音持続時間を示すデュレーションレジスタＤのデータが次に続くことを示すコードであり、デュレーションコードとデュレーションレジスタＤのデータは、必ず組で演奏データメモリに書き込まれる。書き込みを終了した後、演奏データメモリへの書き込みポインタを進めて次の書き込みの準備をする。

【０１０５】ステップＶ８では、発音の終了を示すキーオフコードと演奏データの終了を示すエンドコードを演奏データメモリに書き込む。書き込みを終了した後、演奏データの生成を終了して図２のメインルーチンの処理に戻る。

【０１０６】ステップＶ６において、キーオンフラグＫＯＮが１でないと判断されたときには、現在発音中ではないことを示すので、ステップＶ９へ進み、エンドコードのみを演奏データメモリに書き込む。書き込みを終了した後、図２のメインルーチンの処理へ戻る。

【０１０７】図１１は、図１０のステップＶ４における演奏データ変換処理の詳細を示すフローチャートである。メモリＭＥＭ（Ｐ）にエンドコードではなく、キーコードが格納されているときに、メモリＭＥＭ（Ｐ）のキーコードを基にして演奏データに変換する手順を示す。

【０１０８】ステップＷ１では、メモリＭＥＭ（Ｐ）にオール１が格納されているか否かを調べる。オール１が格納されていれば、入力が無音状態であることを示す。メモリＭＥＭ（Ｐ）にオール１が格納されていなけれ

ば、何らかのキーコードが記憶されているので、そのキーコードを基にした演奏データを生成するためにステップW２へ進む。

【０１０９】ステップW２では、キーオンフラグKONが１であるか否かを調べる。キーオンフラグKONが１でなければ、今までキーオフで発音状態でなかったので、ステップW６へ進み、キーオンフラグKONを１にして、ステップW７へ進む。

【０１１０】ステップW７では、前回のキーオフを示す休符の長さを示すデュレーションコードとデュレーションレジスタDの値を書き込む。書き込みを終了した後、演奏データメモリの書き込みポインタを進めて、ステップW８へ進む。

【０１１１】ステップW８では、演奏データメモリに今回のキーオンを示すキーオンコードとメモリMEM（P）に格納されているキーコードを書き込む。書き込みを終了した後、演奏データメモリの書き込みポインタを進めて、ステップW９へ進む。

【０１１２】ステップW９では、演奏データメモリに書き込んだメモリMEM（P）のキーコードをレジスタKCに格納する。レジスタKCに格納されたキーコードは、次の配列番号P＋１の処理おいて、同じキーコードが連続しているか否かの判断に用いられる。

【０１１３】ステップW１０では、今回演奏データメモリにキーオンコードと共に書き込まれたキーコードの持続時間をカウントし直すために、デュレーションレジスタDを０にリセットする。その後、図１０の処理へ戻り、次の配列番号P＋１についての演奏データの生成処理を行う。

【０１１４】次に、前回演奏データメモリに書き込んだキーコードと同じキーコードがメモリMEM（P）に格納されている場合について説明する。つまり、メモリMEM（P）のキーコードとMEM（P－１）のキーコードが同じ場合である。

【０１１５】その場合、ステップW１において、メモリMEM（P）にオール１が格納されていないと判断され、さらにステップW２において、前回から引き続き今回もキーオン中であると判断され、ステップW３へ進む。

【０１１６】ステップW３では、メモリMEM（P）に格納されているキーコードとレジスタKCに格納されているキーコードが同じであるか否かを調べる。レジスタKCには、前回から引き続きキーオン中であるキーコードが格納されている。

【０１１７】メモリMEM（P）のキーコードとレジスタKCのキーコードが同じであれば、今回の配列番号Pについても前回と同じキーコードを引き続き発音することを示すので、演奏データメモリには何も書き込まずに、図１０の処理へ戻り、デュレーションレジスタDの値のみをインクリメントして、キーコードの発音持続時間をカウントする。

【０１１８】次に、前回演奏データメモリに書き込んだキーコードと異なるキーコードがメモリMEM（P）に格納されている場合について説明する。つまり、キーオン中において、発音すべきキーコードが変化する場合である。

【０１１９】その場合、ステップW１において、メモリMEM（P）にオール１が格納されていないと判断され、さらにステップW２において、前回から引き続き今回もキーオン中であると判断され、ステップW３へ進む。

【０１２０】ステップW３では、メモリMEM（P）に格納されているキーコードとレジスタKCに格納されているキーコードが同じであるか否かを調べる。メモリMEM（P）のキーコードとレジスタKCのキーコードが同じでなければ、今回の配列番号Pについての発音指示は前回のものとは異なるキーコードの発音を意味するので、ステップW４へ進む。

【０１２１】ステップW４では、演奏データメモリに前回までのキーコードの発音の持続時間を示すデュレーションレジスタDの値をデュレーションコードと共に書き込む。書き込みを終了した後、演奏データメモリの書き込みポインタを進めて、次の書き込みに備える。

【０１２２】ステップW５では、前回のキーコードについての発音終了を示すキーオフコードを演奏データメモリに書き込み、演奏データメモリの書き込みポインタを進める。

【０１２３】ステップW８では、今回のキーコードについての発音開始を示すキーオンコードとメモリMEM（P）に格納されているキーコードを演奏データメモリに書き込み、演奏データメモリの書き込みポインタを進める。

【０１２４】その後、ステップW９においてメモリMEM（P）に格納されているキーコードをレジスタKCに格納し、ステップW１０においてデュレーションレジスタDを０にリセットし、次の発音持続時間をカウントし直す。そして、図１０の処理に戻り、次の配列番号P＋１についての処理を行う。

【０１２５】次に、メモリMEM（P）にキーコードでななく無音状態（休符）を示すオール１が格納されている場合の演奏データ生成手順を説明する。その場合、ステップW１において、メモリMEM（P）にオール１が格納されていると判断され、ステップW１１に進む。

【０１２６】ステップW１１では、キーオンフラグKONが１であるか否かを調べる。キーオンフラグKONが１であれば、キーオフを指示するためにステップW１２へ進み、キーオンフラグKONを０にする。

【０１２７】ステップW１３では、演奏データメモリに前回までのキーコードの発音の持続時間を示すデュレーションコードとデュレーションレジスタDの値を書き込

む。書き込みを終了した後、演奏データメモリの書き込みポインタを進めて、次の書き込みに備える。

【０１２８】ステップＷ１４では、前回のキーコードについての発音終了を示すキーオフコードを演奏データメモリに書き込み、演奏データメモリの書き込みポインタを進める。その後、ステップＷ１０へ進みデュレーションレジスタＤを０にリセットして、キーオフ後の休符の持続時間をカウントし直す。そして、図１０の処理へ戻り、次の配列番号Ｐ＋１についての処理を行う。

【０１２９】メモリＭＥＭ（Ｐ）にオール１が格納されていると判断した後において、ステップＷ１１でキーオンフラグＫＯＮが１でないと判断されたときには、キーオフの状態が前回から引き続いていることを示すので、演奏データメモリに何も書き込まずに図１０の処理に戻り、キーオフ中である休符の持続時間を示すデュレーションレジスタＤをインクリメントし、次の配列番号Ｐ＋１についての処理を行う。

【０１３０】以上、演奏データの生成手順を述べたが、演奏データの形式は上述の形式に限られず、ＭＩＤＩ形式等の演奏データを生成するようにしてもよい。本実施例は、マイクから入力される音声信号に対して、量子化処理を行った後に平均ピッチを検出することにより、ピッチの変動に対して悪影響を受けることなく、正しいピッチを検出することができる。

【０１３１】また、検出されたピッチに対して、調検出により得られる調を用いて、誤認識により検出されたピッチの修正を行う。調によって決まる音楽規則を利用して、ピッチの修正処理を行うことにより、認識率の高いピッチ検出を実現することができる。検出修正されたピッチは、演奏データ変換処理を行うことにより、発音可能な演奏データに変換される。

【０１３２】なお、本実施例では、入力された音声から調を検出したが、スイッチ等を用いて演奏者が調を入力するようにしてもよい。以上実施例に沿って本発明を説明したが、本発明はこれらに制限されるものではない。例えば、種々の変更、改良、組合わせ等が可能なことは当業者に自明であろう。

【０１３３】

【発明の効果】入力された音声信号から検出した複数の時間区間毎のピッチについての平均ピッチを求めることにより、ピッチの細かな変動による悪影響を減らすことができる。平均化されたピッチを基にして音高情報を生成することにより、正しい音高情報を得ることができる。

【０１３４】また、楽曲を構成する入力音声信号から検出された音高情報を基にして調の検出を行った後に、検出された調の音楽規則に応じて音高情報の修正を行う。音高情報の修正を行うことにより、正しい音高情報を得ることができる。

【図面の簡単な説明】

【図１】　本発明を実施するための音声信号変換装置のシステム構成例を示すブロック図である。

【図２】　入力変換処理のメインルーチンを示すフローチャートである。

【図３】　図２のステップＰ３における音声入力処理と平均ピッチ検出処理の詳細を示すフローチャートである。

【図４】　ＣＰＵが行う割り込み処理を示すフローチャートである。

【図５】　メモリＭＥＭ（）の構成例を示す概念図である。

【図６】　図２のメインルーチン中のステップＰ６で行うキーコード変換の処理の詳細を示すフローチャートである。

【図７】　図２のメインルーチン中のステップＰ７の調内キーコードへの修正処理の詳細を示すフローチャートである。

【図８】　データメモリに格納されている調の構成音の音名データを示す概念図である。

【図９】　図７のステップＴ２の調内音丸め込み処理の詳細を示すフローチャートである。

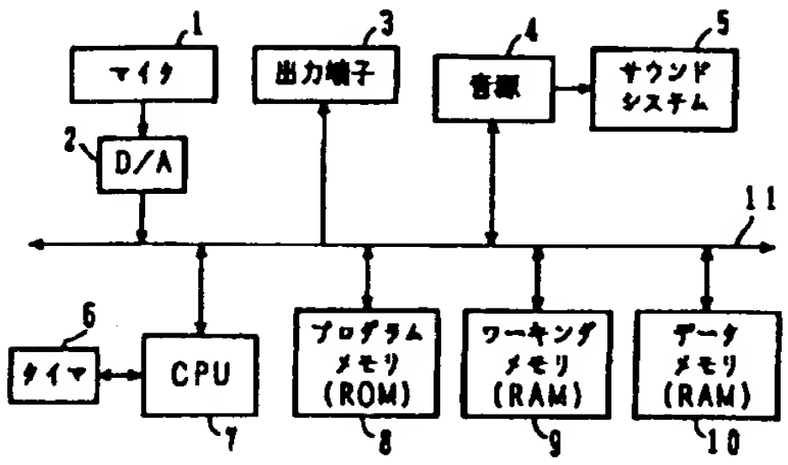
【図１０】　図２のメインルーチン中のステップＰ８の演奏データ変換処理の詳細を示すフローチャートである。

【図１１】　図１０のステップＶ４における演奏データ変換処理の詳細を示すフローチャートである。

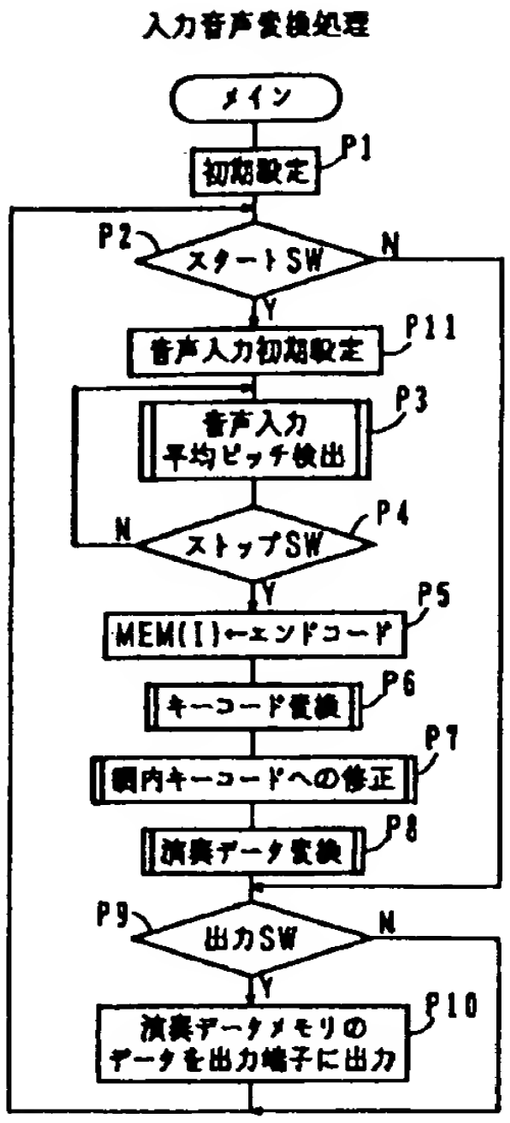
【符号の説明】

１　マイク、　２　Ｄ／Ａ変換器、　３　出力端子、　４　音源、５　サウンドシステム、　６　タイマ、　７　ＣＰＵ、　８　プログラムメモリ、　９　ワーキングメモリ、　１０　データメモリ、　１１　バス

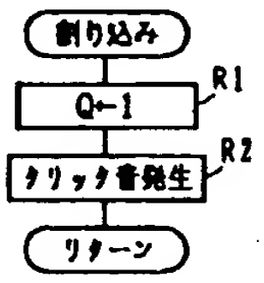
【図1】



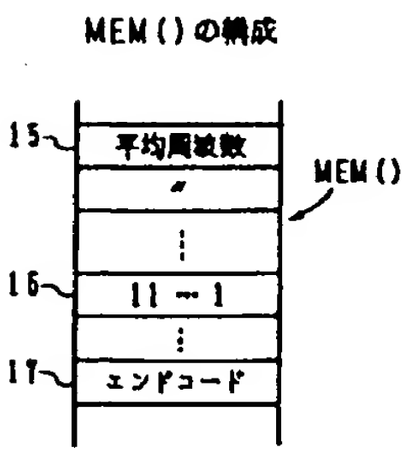
【図2】



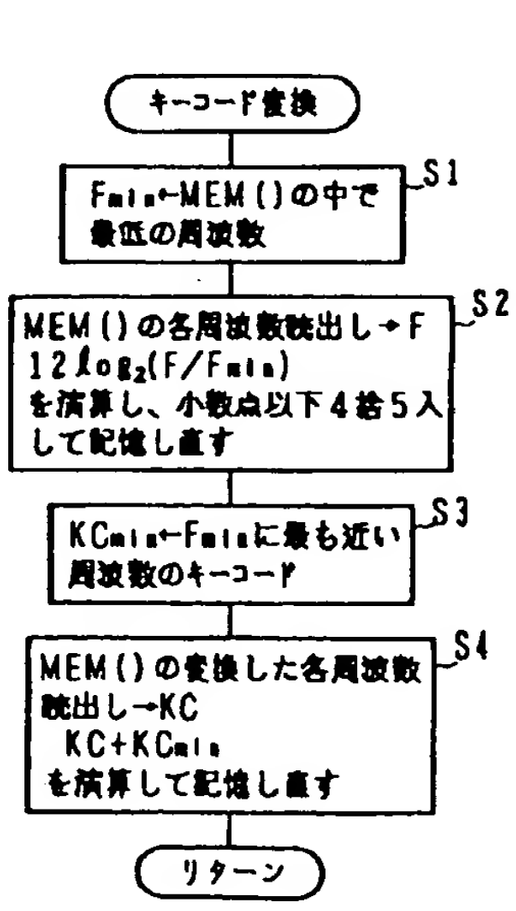
【図4】



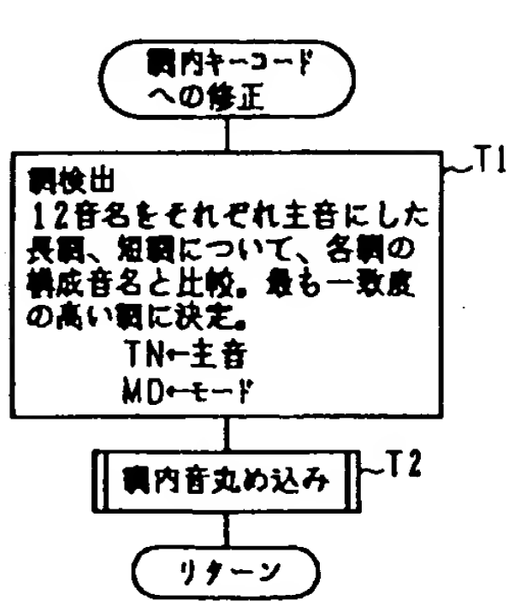
【図5】



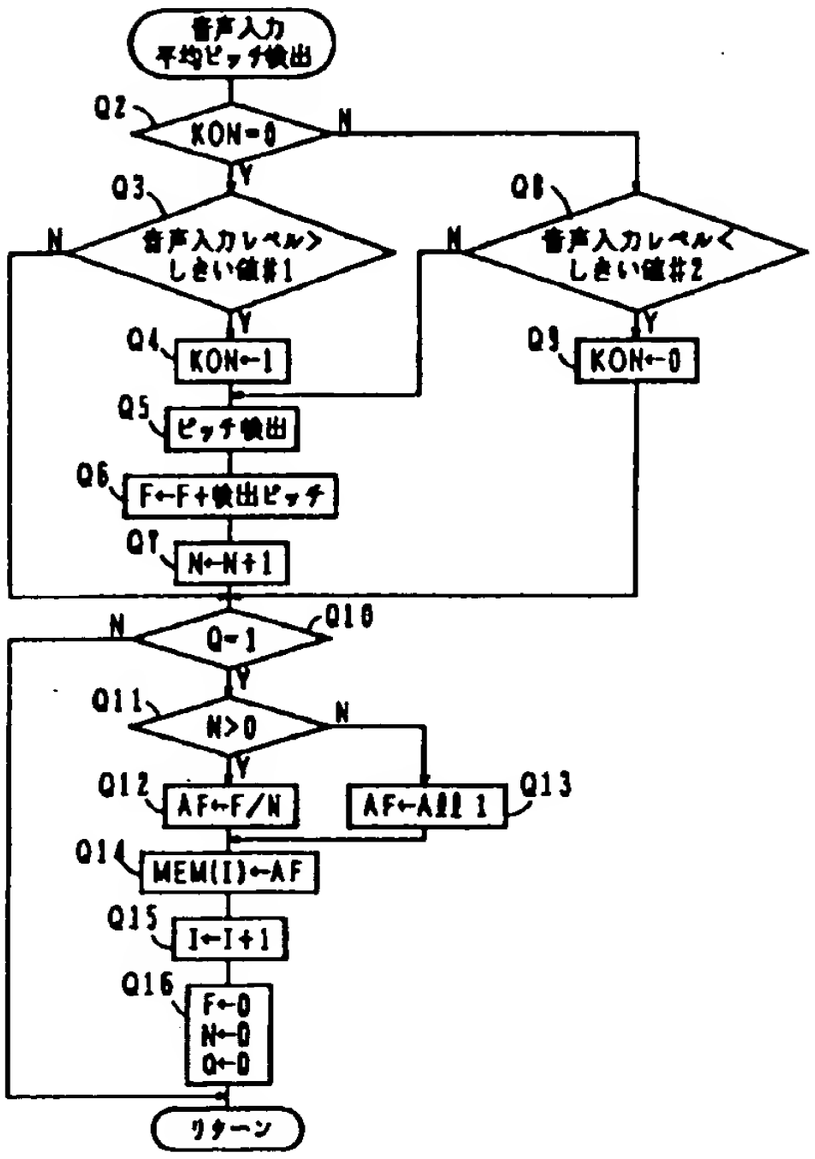
【図6】



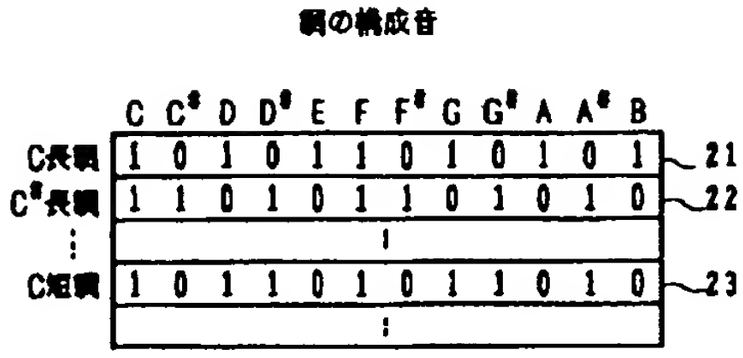
【図7】



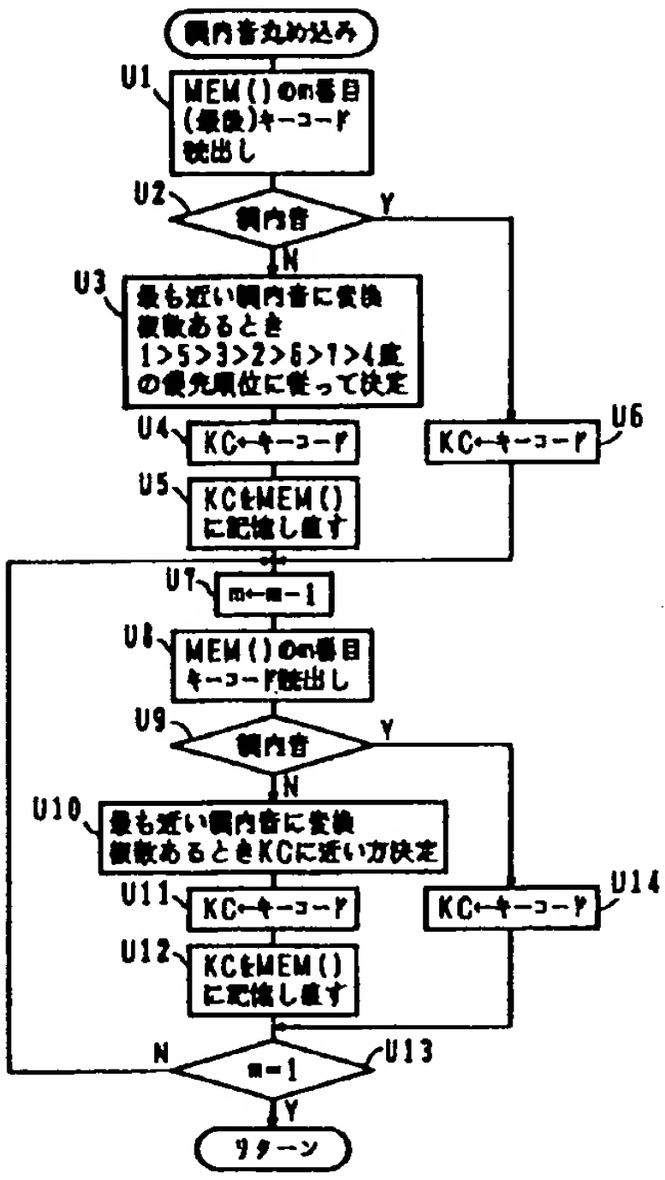
【図３】


音声入力
平均ピッチ検出
Q2
KON=0
N
Y
Q3
音声入力レベル>
しきい値#1
Q8
音声入力レベル<
しきい値#2
Q4
KON←1
Q9
KON←0
Q5
ピッチ検出
Q6
F←F+検出ピッチ
Q7
N←N+1
Q10
Q=1
Q11
N>0
Q12
AF←F/N
AF←ALL 1
Q13
Q14
MEM(I)←AF
Q15
I←I+1
Q16
F←0
N←0
Q←0
リターン

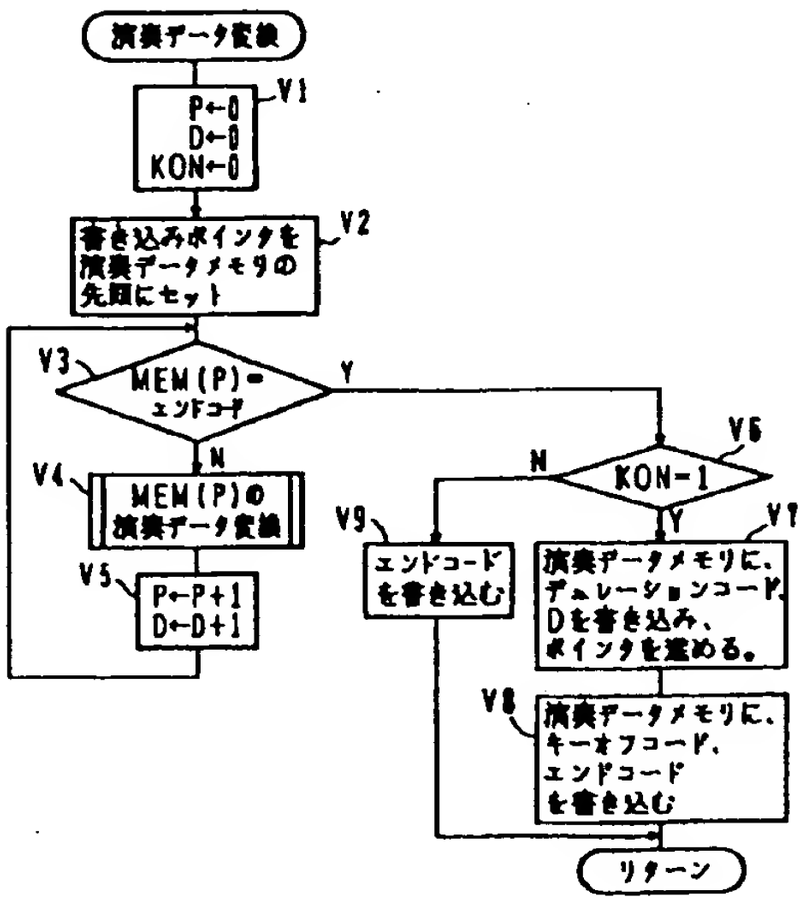
【図８】


調の構成音
C C# D D# E F F# G G# A A# B
C長調 1 0 1 0 1 1 0 1 0 1 0 1
C#長調 1 1 0 1 0 1 1 0 1 0 1 0
C短調 1 0 1 1 0 1 0 1 1 0 1 0
21
22
23

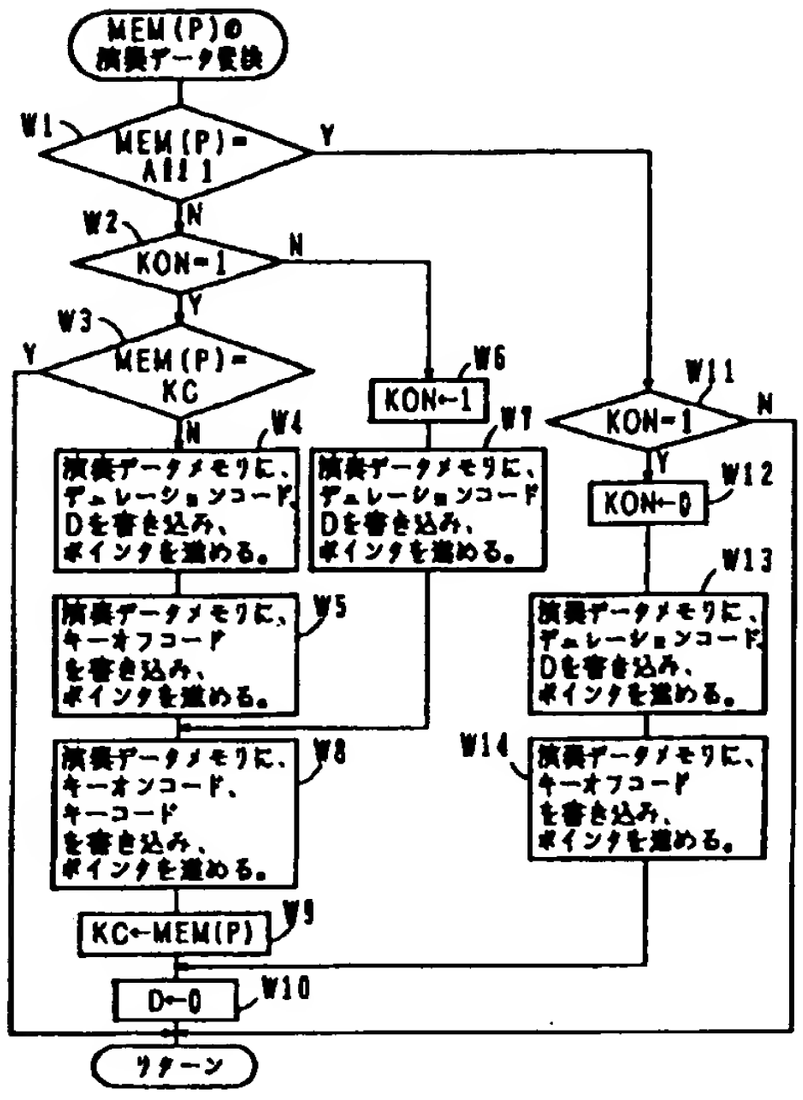
【図９】


調内音丸め込み
U1
MEM()のm番目
(最後)キーコード
読出し
U2
調内音
Y
N
U3
最も近い調内音に変換
複数あるとき
1>5>3>2>6>1>4度
の優先順位に従って決定
U4
KC←キーコード
U5
KCをMEM()
に記憶し直す
U6
KC←キーコード
U7
m←m-1
U8
MEM()のm番目
キーコード読出し
U9
調内音
Y
N
U10
最も近い調内音に変換
複数あるときKCに近い方決定
U11
KC←キーコード
U12
KCをMEM()
に記憶し直す
U14
KC←キーコード
N
m=1
U13
Y
リターン

【図１０】


演奏データ変換
V1
P←0
D←0
KON←0
V2
書き込みポインタを
演奏データメモリの
先頭にセット
V3
MEM(P)=
エンドコード
Y
N
V4
MEM(P)の
演奏データ変換
V5
P←P+1
D←D+1
V6
KON=1
N
Y
V9
エンドコード
を書き込む
V7
演奏データメモリに、
デュレーションコード、
Dを書き込み、
ポインタを進める。
V8
演奏データメモリに、
キーオフコード、
エンドコード
を書き込む
リターン

【図１１】


MEM(P)の
演奏データ読込
W1
MEM(P)=
All 1
Y
N
W2
KON=1
N
Y
W3
MEM(P)=
KC
Y
N
W4
演奏データメモリに、
デュレーションコード
Dを書き込み、
ポインタを進める。
W5
演奏データメモリに、
キーオフコード
を書き込み、
ポインタを進める。
W6
KON←1
W7
演奏データメモリに、
デュレーションコード
Dを書き込み、
ポインタを進める。
W8
演奏データメモリに、
キーオンコード、
キーコード
を書き込み、
ポインタを進める。
W9
KC←MEM(P)
W10
D←0
リターン
W11
KON=1
N
Y
W12
KON←0
W13
演奏データメモリに、
デュレーションコード
Dを書き込み、
ポインタを進める。
W14
演奏データメモリに、
キーオフコード
を書き込み、
ポインタを進める。